Firebird 5 Quick Start Guide

IBPhoenix Editors, Mark Rotteveel, Firebird Project Members

0.2; 8 November 2023

Table of Contents

Chapter 1. About this guide

The *Firebird 5 Quick Start Guide* is an introduction for the complete newcomer to a few essentials for getting off to a quick start with a Firebird binary kit. The guide first saw the light as Chapter 1 of the *Using Firebird* manual, sold on CD by IBPhoenix. Later it was published separately on the Internet. In June 2004, IBPhoenix donated it to the Firebird Project. Since then, it is maintained, and regularly updated, by members of the Firebird documentation project.

If you notice anything missing, unclear, or outright incorrect in this document, please report it on the firebird-documentation issue tracker or submit a pull request fixing the issue.

> Before you read on, verify that this guide matches your Firebird version. This document covers Firebird 5. For all other Firebird versions, get the corresponding Quick Start Guide at https://www.firebirdsql.org/en/documentation/.

Chapter 2. The Firebird licenses

Firebird is a free, open-source database management system, but "free" does not mean that everything is permitted. The use of Firebird is governed by two licenses: the IPL (InterBase Public License) and the IDPL (Initial Developer's Public License). The first one covers the parts of the source code that were inherited from InterBase; the second applies to the additions and improvements made by the Firebird Project. Both licenses offer similar rights and restrictions. In short:

- Use of the software is free, even for commercial purposes. You may also redistribute the software, separately or with a product of your own, but you may not claim ownership or credit for it. Any license notices included with Firebird must remain intact.

- You may modify and recompile the Firebird source code or parts of it. You may distribute such modified versions, but if you do so, you *must* document your modifications and make them publicly available, at no cost, under the same license as the original code.

- You may include Firebird source code (modified or not) in a larger work and distribute that larger work, in source and/or compiled form, under a license of your own choosing. You need not publicize the source code for the entire larger work, but you *must* fulfill the license conditions for the parts that were taken from Firebird, whether they were modified or not.

Please notice that the above is a simplified overview. Only the original license texts are legally binding. You can find them here:

https://www.firebirdsql.org/ipl/ (IPL)
https://www.firebirdsql.org/idpl/ (IDPL)

Chapter 3. Installing Firebird

The instructions given below for the installation of Firebird on Windows and Linux should be sufficient for the vast majority of cases. However, if you experience problems or if you have special needs not covered here, be sure to read the Release Notes. This is especially important if you are upgrading from a previous version or if there are remnants of an old (and maybe long gone) InterBase or Firebird installation floating around your system (DLLs, Registry entries, environment variables…)

3.1. Installation kits

At the Firebird website, https://firebirdsql.org, the installation kits have names like:

`Firebird-5.0.0.bbbb-p-windows-x64.exe`

Windows executable installer, 64 bits

`Firebird-5.0.0.bbbb-p-windows-x64.zip`

Windows zip kit for manual installation

`Firebird-5.0.0.bbbb-p-windows-x86.exe`

Windows executable installer, 32 bits

`Firebird-5.0.0.bbbb-p-linux-x64.tar.gz`

Linux compressed tarball

`Firebird-5.0.0.bbbb-p-macos-x64.pkg`

Mac OS-X 64-bit package

…where `bbbb` is the build number (e.g. 1227 for Firebird 5.0.0 Release Candidate 1) and `p` the packaging number (usually 0 or another low one-digit number).

Firebird 5 packages will also undoubtedly wind up in various Linux distributions and their online repositories. These will have their own naming schemes.

3.2. Installing the Firebird server

3.2.1. Before installation

> If you're installing Firebird on a system where Firebird has never been installed, you can skip this section.

It is almost always advisable to uninstall any previous Firebird installations completely (*after* you've read the next paragraph!) and also hunt the Windows system dirs for old copies of `gds32.dll` and `fbclient.dll`. If you're using Linux, the uninstall scripts should have removed any copies and/or symlinks in `/usr/lib[64]`, but it won't hurt to look if anything named `libfbclient.*` or `libgds.*` is still lying around.

Furthermore, you should be aware that Firebird 5 can open Firebird 4 databases, and even upgrade them, from ODS 13.0 to ODS 13.1, but won't open databases that were created by Firebird 3 or older. So, before taking down your existing setup, you should back up all your databases in order that you can restore them later under Firebird 5.

Upgrading the security database from Firebird 3 and higher

1. Backup your `security3.fdb` or `security4.fdb`

   ```
   gbak -user sysdba -b {path}security4.fdb security4.fbk
   # or using the security.db alias
   gbak -user sysdba -b security.db securitydb.fbk
   # or using XNET (Windows only)
   gbak -user sysdba -pas masterkey -b xnet://security.db securitydb.fbk
   ```

   > You can only perform a local (embedded or XNET) backup of the security database; embedded mode requires filesystem access rights to the database file; XNET (Windows only) requires username **and** password.

2. (only after backing up *all* your databases) Install Firebird 5 (see later sections)

3. Note down the permissions, ownership and filesystem access rights of `security5.fdb`

4. Rename the existing `security5.fdb` for safekeeping

5. Restore the backup of the first step as the security database

   ```
   gbak -user sysdba -c security4.fbk {path}security5.fdb
   # or using the security.db alias
   gbak -user sysdba -c securitydb.fbk security.db
   ```

   Contrary to the backup scenario, using XNET is not an option to restore the security database (as this would require the security database to exist, and it would lock the file).

6. Change filesystem access rights/permissions to match the original `security5.fdb`

For a Firebird 4 security database, it is also possible to copy the `security4.fdb` to `security5.fdb` and use `gfix -upgrade` to upgrade from ODS 13.0 to ODS 13.1, but we recommend using the backup and restore route (if only so you have a backup as a fallback).

Upgrading the security database from Firebird 2.5 and earlier

Back up your old security database `security2.fdb`. Firebird 5 comes with a SQL script `security_database.sql` (located in `misc/upgrade/v3.0`) that will recreate users in the Firebird 5 security database, preserving all information *except* SYSDBA's and except any passwords. For more information, see `doc/README.security_database.txt` in your Firebird 5 installation directory, or *Compatibility Issues* :: *Upgrading a v.2.x Security Database* in the Firebird 3.0 Release Notes.

3.2.2. Installation drives

The Firebird server — and any databases you create or connect to — must reside on a hard drive that is physically connected to the host machine. You cannot locate components of the server, or any database, on a mapped drive, a filesystem share or a network filesystem. (Well, you can, but you shouldn't, and this technique isn't covered here.)

> **ℹ** You can open a read-only database from a read-only medium like a DVD, but you cannot run Firebird server from one.

3.2.3. Installation script or program

Although it is possible to install Firebird by a filesystem copying method — such as untarring a snapshot build or decompressing a `.zip` archive — it is strongly recommended that you use the distributed release kit (`.exe` for Windows, `.tar.gz` for Linux), especially if this is the first time you install Firebird. The Windows installation executable, and the `install.sh` script in the official `.tar.gz` for various POSIX platforms all perform some essential setup tasks. Provided you follow the installation instructions correctly, there should be nothing for you to do upon completion but log in and go!

3.2.4. Server modes

Some installers ask you to choose between Classic, SuperClassic and SuperServer mode. What are they?

- Classic mode (aka *MultiProcess*) involves a single listening process that spawns off an additional process for each client connection. Using a locking mechanism, it allows shared connections to database files.

- SuperClassic (*ThreadedShared*) is a single server process. Client connections are handled by separate threads, each having their own database page cache. Other processes (e.g. embedded servers) may open the same database simultaneously (hence the *Shared*).

- Superserver (*ThreadedDedicated*) is also a single server process with threads handling client connections. There is a single, common database page cache. The server requires exclusive access to each database file it opens (hence the *Dedicated*).

Each mode is fully stable and there is no reason to categorically prefer one to the other. Of course, you may have your own specific considerations. When in doubt, just follow the installer default for now. Changing the server mode later can be done via the configuration file `firebird.conf` and requires a restart but not reinstallation. The server mode can even be configured per database (consult the Firebird 3.0 Release Notes for details).

3.2.5. Installing on Windows

The installer requires Administrator privileges, and Windows will prompt you for privilege elevation when run.

On Windows Server platforms Firebird will run as a system service by default, but during installation you can also choose to let it run as an application. Don't do this unless you have a

compelling reason. If you do want to run Firebird as an application, it is recommended to install it outside the *Program Files* folder to ensure you have sufficient access rights.
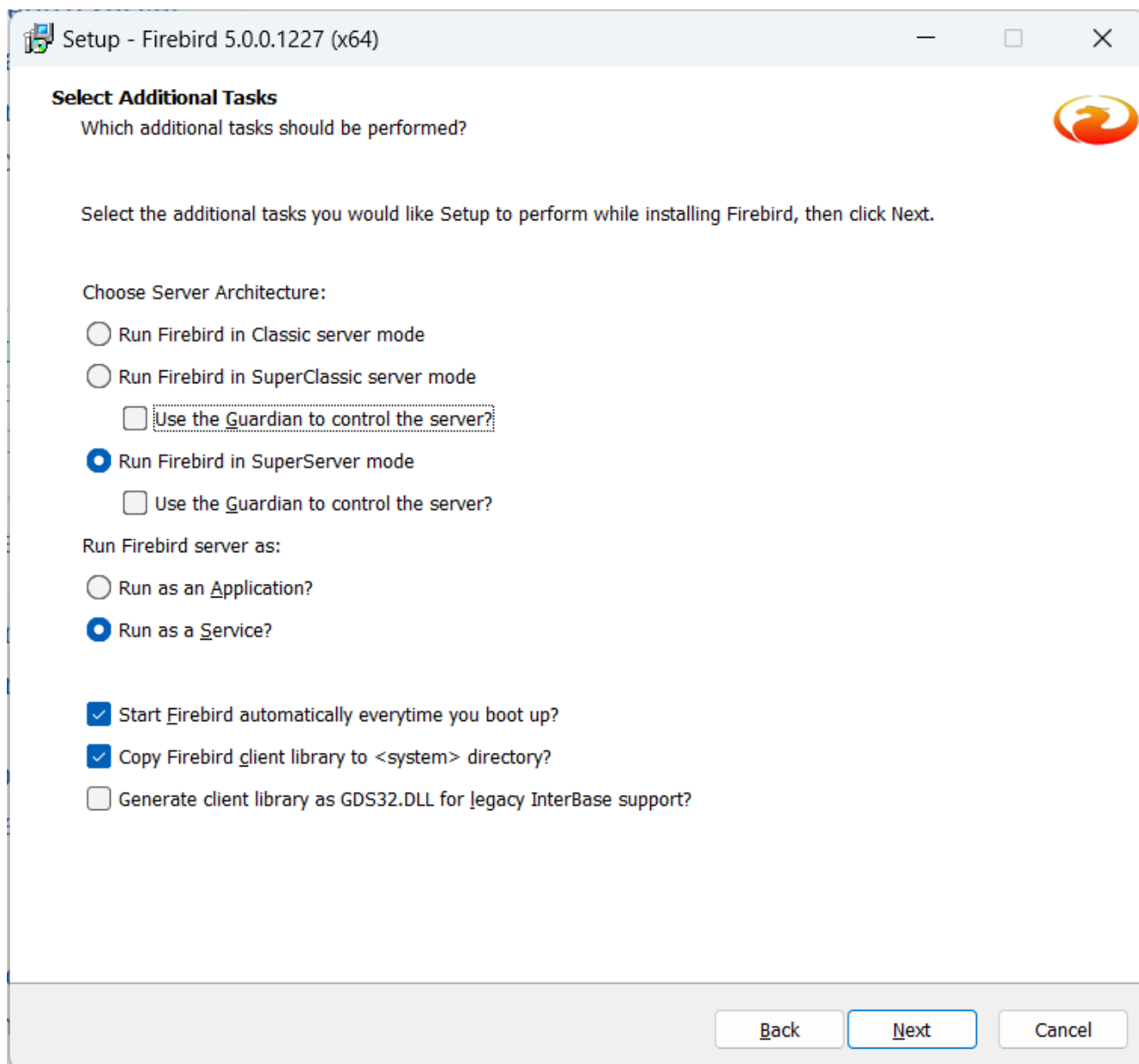
During installation, you have the option of providing a password for Firebird's superuser, SYSDBA. Firebird passwords may be up to 255 bytes long, but due to the nature of the hashing algorithm used by the Srp plugins the "effective length" is around 20 bytes, so it's not very useful to enter a password that's much longer than that. Notice however that if you do enter such a password, you must supply it in its full length every time you connect — it won't work if you truncate it to the first 20 characters!

Legacy authentication

The installer does not offer an option to enable legacy authentication. You will need to configure this manually if you require it. If security is a concern (as it should be), you should not use the Legacy_Auth authentication plugin or enable it only temporarily while you upgrade your existing clients to Firebird 5.0. The legacy connection method sends passwords over the wire unencrypted, does not support wire protocol encryption, and also limits

(truncates!) the usable length of the password to 8 bytes.

Use the Guardian?

The Firebird Guardian is a utility that monitors the server process and tries to restart it if it terminates abnormally. During a Windows install, you can opt to use the Guardian when running in SuperClassic or Superserver mode. However, since Windows has the facility to watch and restart services, there is no reason to use the Guardian if Firebird runs as a service (which it should).

The Guardian may be phased out in future versions of Firebird.

3.2.6. Installing on Linux and other Unix-like platforms

In all cases, read the Release Notes for the Firebird version you're going to install. There may be significant variations from release to release of any POSIX operating system, especially the open source ones. Where possible, the build engineers for each Firebird version have attempted to document any known issues.

Aside from being packaged with the download kits, Release Notes for all officially released versions of Firebird can also be found at https://www.firebirdsql.org/en/release-notes/.

For Linux distributions, use the `.tar.gz` kit. Quite often, installation is just a matter of untarring the archive and running `install.sh`. In some cases, the Release Notes or packed Readmes may instruct you to edit the scripts and make some manual adjustments.

3.3. Installing multiple servers

Firebird allows the operation of multiple servers on a single machine. It can also run concurrently with Firebird 1.x or InterBase servers. Setting this up is not a beginner's task though. If you need to run multiple servers on the same machine, the second and subsequent servers must be installed and configured manually. They need to have different service names and should listen on different TCP/IP ports. The file `install_windows_manually.txt` in the `doc` subdirectory may be of help if you're doing this on Windows, but bear in mind that it was written for Firebird 2.1.

Also read the chapter *Configuring the Port Service on Client and Server* in the Firebird 1.5 (!) Release Notes:

https://www.firebirdsql.org/file/documentation/release_notes/html/rlsnotes15.html#config-port
https://www.firebirdsql.org/file/documentation/release_notes/Firebird-1.5.6-ReleaseNotes.pdf#page=96

3.4. Testing your installation

If you want to connect to your Firebird server across a network, then before testing the Firebird server itself, you may want to verify that the server machine is reachable from the client at all. At this point, it is assumed that you will use the TCP/IP network protocol for your Firebird client/server connections.

3.4.1. Pinging the server

The `ping` command — available on most systems — is a quick and easy way to see if you can connect to a server machine via the network. For example, if your server's IP address in the domain that is visible to your client is 192.13.14.1, go to a command shell on the client machine and type the command

```
ping 192.13.14.1
```

substituting this example IP address with the IP address that your server is broadcasting. If you are on a managed network, and you don't know the server's IP address, ask your system administrator. Of course, you can also ping the server by its name, if you know it:

```
ping vercingetorix
```

If you are connecting to the server from a local client — that is, a client running on the same machine as the server — you can ping the virtual TCP/IP loopback:

```
ping localhost
```

or

```
ping 127.0.0.1
```

If you have a simple network of two machines linked by a crossover cable, you can set up your server with any IP address you like except 127.0.0.1 (which is reserved for a local loopback) and, of course, the IP address which you are using for your client machine. If you know the "native" IP addresses of your network cards, and they are different, you can simply use those.

Once you have verified that the server machine is reachable from the client, you can go on to the next step.

3.4.2. Making sure that the Firebird server is running

Most — but not all — installation packages start up the Firebird server as one of the final steps during installation, and also make sure that Firebird is started at every reboot.

After being launched, the Firebird server should be running:

On Linux or other Unix-like systems

As a service.

On Windows server systems

As a service or as an application. Service is default and highly recommended.

The following sections show you how to test the server on each platform.

Server check: Linux and other Unices

Use the `top` command in a command shell to inspect the running processes interactively. If a Firebird 5 server is running, you should see a process named `firebird` and possibly also `fbguard` (the Guardian process).

The following screen shows the output of `top`, restricted by `grep` to show only lines containing the string `firebird`:

```
paul@fili ~ $ top -b -n1 | grep [f]irebird
 7169 firebird  20   0   29668    992    560 S   0,0 0,0   0:00.00 fbguard
 7171 firebird  20   0  228160   5876   3048 S   0,0 0,1   0:00.01 firebird
```

As an alternative to `top`, you can use `ps -ax` or `ps -aux` and pipe the output to `grep`.

The process name is `firebird` regardless if Firebird is running in Superserver, Classic or SuperClassic mode. However, it is possible to configure a Classic-mode Firebird in such a way that it runs as a service under (x)inetd. In that case, you will only see a `firebird` process if a client connection has been made.

Another way of testing the server after installation is by starting a Firebird client (e.g. `/opt/firebird/bin/isql`) and connecting to a database or creating one. These operations are described later in this guide.

If it turns out that the server hasn't been started after all, you may need to do this manually, e.g. with `/etc/init.d/firebird start` or `systemctl start firebird` and `systemctl enable firebird`, depending on the type of Linux system and your Firebird installation package.

Server check: Windows, running as service

Use Win + R, type `services.msc` followed by Enter or **[OK]**, or click **[Start]** (or Win) and search for "Services" and open the "Services" app.

This illustration shows the Services applet display on Windows 11. The appearance may vary by Windows edition and configuration.
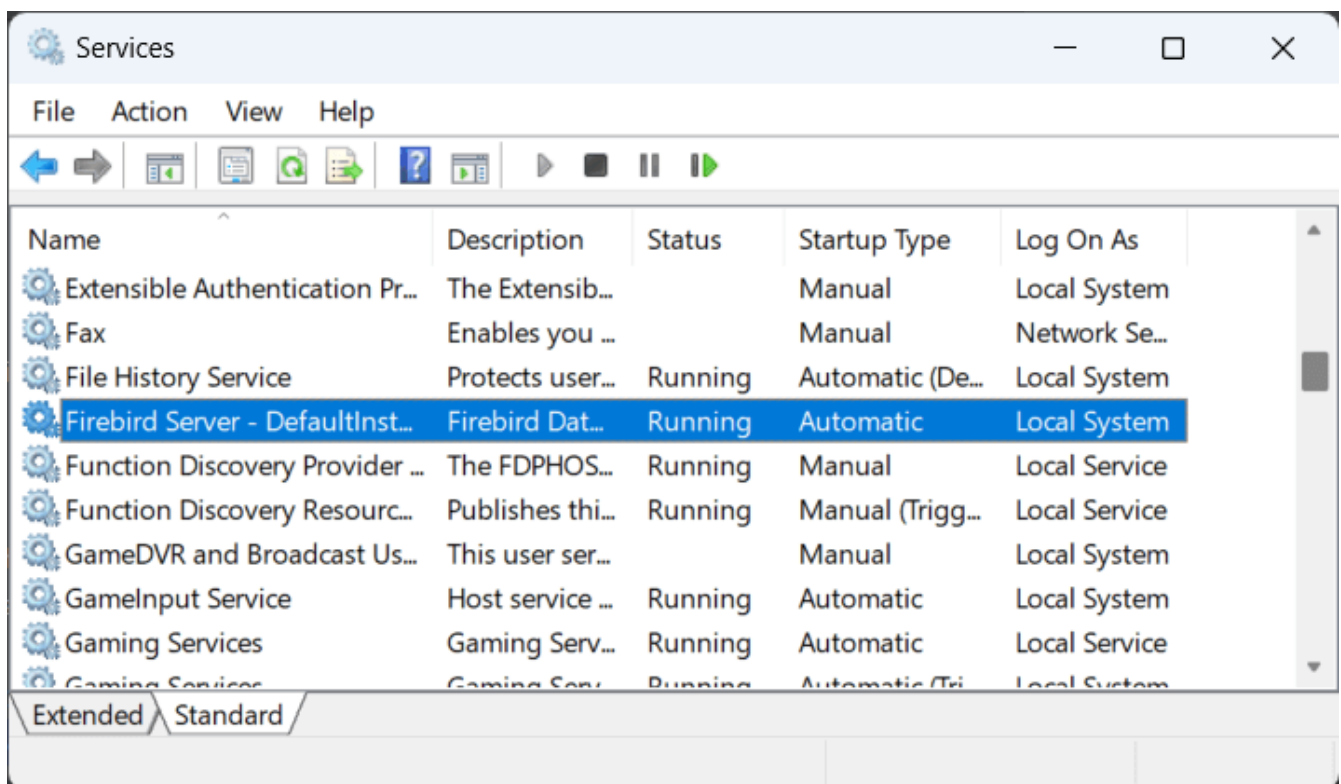
You should at least find the Firebird server in the services listing. The Guardian may or may not be running, depending on the choices you made during installation. If you didn't opt to start the server at the end of the installation process, you may do so now by right-clicking on the Firebird entry (or the Guardian) and choosing Start.

Server check: Windows, running as application

If Firebird is up and running as an application, it is represented by an icon in the system tray:

- A green and grey server symbol if controlled by the Guardian;
- A round yellow and black graphic if running standalone.

A flashing icon indicates that the server is in the process of starting up (or at least trying to do so). A red icon, or an icon with an overlying red stop sign, indicates that startup has failed.

One way to make 100% sure if the server is running or not is to press `Ctrl` + `Alt` + `Del`, select *Task Manager,* and look for the `firebird` process (and possibly `fbguard`) in the task list. You may need to check the box "Show processes of all users" for these processes to become visible.

On some occasions, you may need to start the Guardian or server once explicitly via the Start menu even if you opted for "Start Firebird now" at the end of the installation process. Sometimes a reboot is necessary.

You can shut the server down via the menu that appears if you right-click on the tray icon. Notice that this also makes the icon disappear; you can restart Firebird via the Start menu.

> In Classic mode (but not SuperClassic!) a new process is launched for every connection, so the number of `firebird` processes will always equal the number of client connections plus one. Shutdown via the tray icon menu only terminates the first process (the *listener*). Other processes, if present, will continue to function

normally, each terminating when the client disconnects from the database. Of course, once the listener has been shut down, new connections can't be made.

3.5. Performing a client-only install

Each remote client machine needs to have the client library — `libfbclient.so` on POSIX clients, `fbclient.dll` on Windows clients — that matches the release version of the Firebird server.

Firebird can install symlinks or copies named after the 1.0 libs (with the "old" InterBase names), to maintain compatibility with third-party products which need these files.

Some extra pieces are also needed for the client-only install.

3.5.1. Windows

No separate installation program is available to install only the client pieces on a Windows machine. If you are in the common situation of running Windows clients to a Linux or other Unix-like Firebird server (or another Windows machine), you need to download the full Windows installation kit that corresponds to the version of Firebird server you install on your server machine.

Fortunately, once you have the kit, the Windows client-only install is a breeze. Just run the installation program and when you arrive at the "Select Components" screen, choose one of the client-only options from the drop-down list or uncheck the "Server Components" checkbox.

3.5.2. Linux and some other POSIX clients

A small-footprint client install program for Linux clients is not available either. Additionally, some POSIX flavours — even within the Linux constellation — have somewhat idiosyncratic requirements for filesystem locations. For these reasons, not all *nix distributions for Firebird even contain a client-only install option.

For most Linux flavours, the following procedure is suggested for a manual Firebird client-only install. Log in as `root` for this.

1. Look for `libfbclient.so.5.0.n` (n being the patch version number) in `/opt/firebird/lib` on the machine where the Firebird server is installed. Copy it to `/usr/lib` on the client (or `/usr/lib64` if both server and client are 64-bits).

2. Create chained symlinks using the following commands:

   ```
   ln -s /usr/lib/libfbclient.so.5.0.n /usr/lib/libfbclient.so.2

   ln -s /usr/lib/libfbclient.so.2 /usr/lib/libfbclient.so
   ```

   ...replacing `5.0.n` with your version number, e.g. `5.0.0` or `5.0.1`

   If you're running applications that expect the legacy libraries to be present, also create the following symlinks:

```
ln -s /usr/lib/libfbclient.so /usr/lib/libgds.so.0

ln -s /usr/lib/libfbclient.so /usr/lib/libgds.so
```

3. Copy `firebird.msg` to the client machine, preferably into the `/opt/firebird` directory. If you place it somewhere else, create a system-wide permanent `FIREBIRD` environment variable pointing to the right directory, so that the API routines can locate the messages.

4. Optionally copy the Firebird command-line tools — e.g. `isql` — to the client machine as needed.

Instead of copying the files from a server, you can also pull them out of a Firebird `tar.gz` kit. Everything you need is located in the `/opt/firebird` tree within the `buildroot.tar.gz` archive that's packed inside the kit.

Chapter 4. Default disk locations

The tables below show you where you'll find the Firebird files and directories after a standard installation. Please notice that the listings are not exhaustive.

4.1. Linux

The following table shows the default component locations of a Firebird installation on Linux. Some locations may be different on other Unix-like systems, or on certain Linux distributions.

Table 1. Firebird 5.0 component locations on Linux

Component	File Name	Default Location
Installation directory (referred to hereafter as `$(install)`)	-	`/opt/firebird` (may vary per distribution)
Configuration files	`firebird.conf, databases.conf`, etc.	`$(install)`
Release Notes and other documentation	Various files	`$(install)/doc`
Firebird server	`firebird`	`$(install)/bin`
Command-line tools	`isql, gbak, nbackup, gfix, gstat`, etc.	`$(install)/bin`
Plugins	`libEngine13.so, libSrp.so, libudr_engine.so`, etc.	`$(install)/plugins`
Sample database	`employee.fdb`	`$(install)/examples/empbuild`
Additional server-side libraries	`libib_util.so`	`$(install)/lib`
Client libraries	`libfbclient.so.5.0.n` The usual symlinks (`*.so.2, *.so`) are created. Legacy `libgds.*` symlinks are also installed.	`/usr/lib[64]` (actually, the real stuff is in `$(install)/lib`, but you should use the links in `/usr/lib[64]`)

4.2. Windows

In the table below, `%ProgramFiles%` refers to the Windows programs folder. This is usually "`C:\Program Files`" but may also be a different path, e.g. "`D:\Programmi`". Likewise, `%windir%` refers to the Windows directory. Be sure to read the notes below the table, especially if you're running Firebird on a 64-bit Windows system.

Table 2. Firebird 5.0 component locations on Windows

Component	File Name	Default Location
Installation directory (referred to hereafter as `$(install)`)	-	`%ProgramFiles%\Firebird\Firebird_5_0`
Configuration files	`firebird.conf`, `databases.conf`, etc.	`$(install)`
Release Notes and other documentation	Various files	`$(install)\doc`
Firebird server	`firebird.exe`	`$(install)`
Command-line tools	`isql.exe`, `gbak.exe`, `nbackup.exe`, `gfix.exe`, `gstat.exe`, etc.	`$(install)`
Plugins	`engine13.dll`, `srp.dll`, `udr_engine.dll`, etc.	`$(install)\plugins`
Sample database	`employee.fdb`	`$(install)\examples\empbuild`
Internationalisation	`fbintl.conf`, `fbintl.dll`	`$(install)\intl`
Additional server-side libraries	`icu*.dll`, `ib_util.dll`	`$(install)`
Client connection libraries	`fbclient.dll` (with an optional `gds32.dll`, to support legacy apps)	`$(install)` (with an optional copy in `%windir%\System32` — see note below table)
Some necessary Microsoft runtime libs	`msvcp140.dll`, `vcruntime140.dll`	`$(install)`
32-bit library versions for use with 64-bit Firebird	`fbclient.dll`, `msvcp140.dll`, `vcruntime140.dll`	`$(install)\WOW64` (with an optional copy in `SysWOW64` — see second note below table)

The Windows directories

A typical location for the Windows system directory — on both 32-bit and 64-bit systems — is `%windir%\System32`, e.g. `C:\Windows\System32`.

On 64-bit Windows systems, the "`Program Files`" directory is reserved for 64-bit programs. If you try to install a 32-bit application into that folder, it will be auto-redirected to a directory which — in English versions — is called "`Program Files (x86)`". In other language versions the name may be different.

In the same vein, the `System32` directory is reserved for 64-bit libraries, and 32-bit libraries go into `SysWOW64`. That's right: 64-bit libraries are in `System32`, 32-bit libraries in `SysWOW64`.

If you're not aware of this, you may have a hard time locating your 32-bit Firebird components on a 64-bit Windows system.

(Incidentally, *WOW* stands for *Windows on Windows*.)

Chapter 5. Server configuration and management

There are several things you should be aware of — and take care of — before you start using your freshly installed Firebird server. This part of the manual introduces you to some useful tools and shows you how to protect your server and databases.

5.1. User management

Since Firebird 3, user management is done entirely through SQL commands. Users of previous versions are probably familiar with the `gsec` utility for this task. It is still present, but deprecated, and it won't be discussed here.

5.1.1. Changing the SYSDBA password

One Firebird account is created automatically as part of the installation process: SYSDBA. This account has all the privileges on the server and cannot be deleted. Depending on version, OS, and architecture, the installation program will either

- install the SYSDBA user with the password `masterkey`, or

- ask you to enter a password during installation, or

- generate a random password and store that in the file `SYSDBA.password` within your Firebird installation directory.

If the password is `masterkey` and your server is exposed to the Internet at all — or even to a local network, unless you trust every user with the SYSDBA password — you should change it immediately. Fire up `isql` or another Firebird client and connect to a database. In this example, the "employee" example database is used, because its alias is always present in a freshly installed Firebird setup:

```
connect localhost:employee user sysdba password masterkey;
```

If you do this in `isql`, it should respond with:

```
Database: localhost:employee, User: SYSDBA
```

Now alter the sysdba password:

```
alter user sysdba set password 'Zis4_viZuna83YoYo';
```

> Instead of USER SYSDBA you can also use CURRENT USER, which always refers to the user you are logged in as.

If the command succeeds, you won't get any feedback. Instead, `isql` will just print the next "SQL>"-

prompt, thus indicating that all is well and your further input is awaited.

Please notice that unlike "regular" usernames, Firebird passwords are always case-sensitive.

> Depending on the `UserManager` **and** `AuthServer` setting in `firebird.conf` or `databases.conf`, you may have **two or more** SYSDBA accounts. Make sure to change the password for all of them.
>
> If you have enabled legacy authentication, you may want to delete the legacy SYSDBA account, as it is insecure and restricts the maximum password length to 8 characters.

5.1.2. Adding Firebird user accounts

Firebird allows the creation of many different user accounts. Each of them can own databases and also have various types of access to databases and database objects it doesn't own.

Assuming you are connected to a database as SYSDBA, you can add a user account as follows:

```
create user billyboy password 'TooLongFor8099Comfort';
```

The full range of user management commands is:

```
CREATE USER username
  <user_option> [<user_option> ...]
  [TAGS (<user_var> [, <user_var> ...]]

[CREATE OR] ALTER {USER username | CURRENT USER}
  [SET] [<user_option> [<user_option> ...]]
  [TAGS (<user_var> [, <user_var> ...]]

DROP USER username)
  [USING PLUGIN _plugin_name]

<user_option> ::=
    PASSWORD 'password'
  | FIRSTNAME 'firstname'
  | MIDDLENAME 'middlename'
  | LASTNAME 'lastname'
  | {GRANT | REVOKE} ADMIN ROLE
  | {ACTIVE | INACTIVE}
  | USING PLUGIN plugin_name

<user_var> ::=
    tag_name = 'tag_value'
  | DROP tag_name
```

Tags are optional key-value pairs that can be freely defined by the user. The key (tag name) must be

a valid SQL identifier, the value a non-`NULL` string of at most 255 bytes.

Only `SYSDBA` and co-admins can use all these commands. Ordinary users can change their own parameters (such as password, name parts and tags, but not active/inactive) using `ALTER USER name` or `ALTER CURRENT USER`. It is not possible to change an account name.

Examples:

```
create user dubya password 'Xwha007_noma'
  firstname 'GW' lastname 'Shrubbery';
create user lorna password 'Mayday_domaka'
  tags (Street = 'Main Street', Number = '888');
alter user benny tags (shoesize = '8', hair = 'blond', drop weight);
alter current user set password 'SomethingEvenMoreSecretThanThis';
alter user dubya set inactive;
drop user ted;
```

For details on managing users through SQL, also see the *Firebird 5.0 Language Reference*, section *SQL Statements for User Management*.

5.1.3. The security database

Firebird user accounts are kept in a *security database*, which normally resides in the installation directory and is called `security5.fdb` (alias: `security.db`). Except in the case of so-called embedded connections (more about those later in this guide), connecting to a database always involves the security database, against which the user credentials are verified. Of course this is done transparently; the user doesn't have to make an explicit connection to the security database.

However, in Firebird 5 this is not the end of the story. Firebird — since Firebird 3 — allows the use of multiple security databases on a system, each security database governing a specific set of databases. A database can even act as its own security database.

Showing how to set this up is outside the scope of this Quick Start Guide. You can find full details in the Firebird 3.0 Release Notes, chapter *Security*. But it is important to realise that *if* a system has multiple security databases, managing user accounts while connected to a database will always affect the accounts in the security database that governs *that* specific database. To be on the safe side, you may want to connect to the security database itself before issuing your user management commands. Connecting to the security database used to be forbidden in recent versions of Firebird, but is now once again possible, albeit by default only locally using embedded or — on Windows — XNET (which means that even the `localhost` route is blocked).

5.1.4. Appointing co-administrators

What follows here is not essential knowledge for beginners. You can skip it if you like and go on to the Security section.

In Firebird 2.5 and up, `SYSDBA` (and others with administrator rights) can appoint co-administrators. This is done with the `GRANT ADMIN ROLE` directive:

```
create user bigbill password 'bigsekrit7foryou' grant admin role;
alter user littlejohn grant admin role;
```

The first command creates user `bigbill` as a Firebird administrator, who can add, alter and drop users. The second command grants administrator privileges to the existing user `littlejohn`.

To revoke administrator privileges from an account, use ALTER USER ⋯ REVOKE ADMIN ROLE.

> - GRANT ADMIN ROLE and REVOKE ADMIN ROLE are not GRANT and REVOKE statements, although they look that way. They are clauses of the CREATE and ALTER USER statements. The actual role name involved here is RDB$ADMIN. This role also exists in regular databases; more about that in a minute.
>
> - Every user who has received administrator rights can pass them on to others. Therefore, there is no explicit WITH ADMIN OPTION.
>
> - Just for completeness, administrators can also grant admin rights to an existing user by connecting to the security database and issuing a regular GRANT statement:
>
>   ```
>   grant rdb$admin to littlejohn
>   ```

Differences between co-administrators and SYSDBA

- Co-admins can create, alter and drop users, but contrary to SYSDBA they have no automatic privileges in regular databases.

- Unlike SYSDBA, co-admins must specify the RDB$ADMIN role explicitly if they want to exert their rights as system administrator:

  ```
  connect security.db user bigbill password bigsekrit7foryou role rdb$admin
  ```

For reasons explained elsewhere in this guide, connecting to the security database like this may fail if a Superserver is running. On Windows, you may circumvent this by prepending `xnet://` to the database path or alias, but on POSIX, you're stuck. The only solution there is to connect to a regular database through the server, for example using `localhost:employee`. (This must be a database that uses the security database in question.)

Since Firebird 4.0, a co-admin no longer needs to have the RDB$ADMIN privileges in the regular database to be able to execute user management statements against the security database, their privileges in the security database are sufficient.

> **Please remember**
>
> The RDB$ADMIN role in a database gives the grantee SYSDBA rights *in that database only*!
>
> - If it is the security database, the grantee can manage user accounts, but has

no special privileges in other databases.

- If it is a regular database, the grantee can control that database like they are SYSDBA, but again they have no special privileges in other databases, and have no user administration privileges.

Of course, it is possible to grant a user the RDB$ADMIN role in several databases, including the security database.

- To grant a user admin rights in a regular database you can use the usual way that roles are granted:

```
grant rdb$admin to bigbill
```

Grantors can be the database owner, SYSDBA, and every other user who has the RDB$ADMIN role in that database and has specified it while connecting. Every RDB$ADMIN member in a database can pass the role on to others, so again there is no WITH ADMIN OPTION.

5.2. Security

Firebird 5 offers a number of security options, designed to make unauthorised access as difficult as possible.

It pays to familiarise yourself with Firebird's security-related configuration parameters. You can significantly enhance your system's security if you raise the protection level wherever possible. This is not only a matter of setting parameters, by the way: other measures involve tuning filesystem access permissions, an intelligent user accounts policy, etc.

Below are some guidelines for protecting your Firebird server and databases.

Run Firebird as non-system user

On Unix-like systems, Firebird already runs as user firebird by default, not as root. On Windows server platforms, you can also run the Firebird service under a designated user account (e.g. Firebird). The default practice — running the service as the LocalSystem user — poses a security risk if your system is connected to the Internet. Consult README.instsvc.txt in the doc directory to learn more about this.

Change SYSDBA's password

As discussed before, if your Firebird server is reachable from the network and the system password is masterkey, change it.

Don't create user databases as SYSDBA

SYSDBA is a very powerful account, with full (destructive) access rights to all your Firebird databases. Its password should be known to a few trusted database administrators only. Therefore, you shouldn't use this super-account to create and populate regular databases. Instead, generate normal user accounts and grant them the CREATE DATABASE privilege, and provide their account names and passwords to your users as needed. You can do this with the SQL user management commands as shown above, or with any decent third-party Firebird

administration tool.

Protect databases on the filesystem level

Anybody who has filesystem-level read access to a database file can copy it, install it on a system under their own control, and extract all data from it — including possibly sensitive information. Anybody who has filesystem-level write access to a database file can corrupt it or totally destroy it.

Also, anybody with filesystem-level access to a database can make an embedded connection to it posing as *any* Firebird user (including SYSDBA) without having their credentials checked. This can be especially disastrous if it concerns the security database!

As a rule, only the Firebird server process should have access to the database files. Users don't need, and should not have, access to the files — not even read-only. They query databases via the server, and the server makes sure that users only get the allowed type of access (if at all) to any objects within the database.

As a relaxation of this rule, most Firebird configurations allow users to create and use databases in their own filesystem space and make embedded connections to them. Since these are *their* files and *their* data, one may argue that unrestricted and possibly destructive access should be their own concern, not yours.

If you don't want or need this relaxation, follow the instructions in the next item.

Disable embedded connections

If you don't want any type of direct access, you may disable embedded mode (= direct filesystem-level access) altogether by opening `firebird.conf` and locating the `Providers` entry. The default (which is probably commented out) is:

```
#Providers = Remote,Engine13,Loopback
```

Now, either remove the hash mark and the `Engine13` provider (this is the one that makes the embedded connections), or — better — add an uncommented line:

```
Providers = Remote,Loopback
```

The `Remote` provider takes care of remote connections; the `Loopback` provider is responsible for TCP/IP connections via `localhost`, as well as (on Windows) and XNET connections to databases on the local machine. All these connection types require full authentication and have the server process, not the user process, open the database file.

Please notice that you can also set the `Providers` parameter on a per-database basis. You can set a default in `firebird.conf` as shown above, and then override it for individual databases in `databases.conf` like this:

```
bigbase = C:\Databases\Accounting\Biggus.fdb
{
```

```
   Providers = Engine13,Loopback
 }
```

The first line defines the *alias* (see next item), and everything between the curly brackets are parameters for that specific database. You'll find `databases.conf` in the same directory as `firebird.conf`. Refer to the Release Notes, chapter *Configuration Additions and Changes*, section *Per-database Configuration*, for more information about the various parameters.

Use database aliases

Database aliases hide physical database locations from the client. Using aliases, a client can — for example — connect to "frodo:zappa" without having to know that the real location is `frodo:/var/firebird/music/underground/mothers_of_invention.fdb`. Aliases also allow you to relocate databases while the clients keep using their existing connection strings.

Aliases are listed in the file `databases.conf`, in this format on Windows machines:

```
 poker = E:\Games\Data\PokerBase.fdb
 blackjack.fdb = C:\Firebird\Databases\cardgames\blkjk_2.fdb
```

And on Linux:

```
 books = /home/bookworm/database/books.fdb
 zappa = /var/firebird/music/underground/mothers_of_invention.fdb
```

Giving the alias an `.fdb` (or any other) extension is fully optional. Of course if you do include it, you must also specify it when you use the alias to connect to the database.

Aliases, once entered and saved, take effect immediately. There is no need to restart the server.

Restrict database access

The `DatabaseAccess` parameter in `firebird.conf` can be set to `Restrict` to limit access to explicitly listed filesystem trees, or even to `None` to allow access to aliased databases only. Default is `Full`, i.e. no restrictions.

Note that this is not the same thing as the filesystem-level access protection discussed earlier: when `DatabaseAccess` is anything other than `Full`, the server will refuse to open any databases outside the defined scope even if it has sufficient rights on the database files.

Choose your authentication method(s)

Firebird supports three authentication methods when connecting to databases:

1. *Srp (Secure Remote Password):* The user must identify themselves with a Firebird username and password, which the server checks against the security database. The maximum effective password length is around 20 bytes, although you may specify longer passwords up to 255 characters. Wire encryption is used. The server supports various `Srp` authentication plugins, with `Srp256` as the default (which uses SHA256 for the user proof).

2. *Win_Sspi (Windows Security Support Provider Interface):* The user is logged in automatically with their Windows account name. Wire encryption is used.

3. *Legacy_Auth:* Insecure method used in previous Firebird versions. Passwords have a maximum length of 8 bytes and are sent unencrypted across the wire. Avoid this method if possible. Wire encryption is **not** supported.

Two configuration parameters control Firebird's authentication behaviour:

- `AuthServer` determines how a user can connect to the local server. It defaults to "`Srp256`", or — on Windows — "`Srp256, Win_Sspi`". In the latter case, the user will be authenticated with their Windows login if they fail to supply user credentials (causing the `Srp256` method, which is tried first, to fail).

- `AuthClient` defines how the local client tries to authenticate the user when making a connection. It is usually "`Srp256, Srp, Win_Sspi, Legacy_Auth`", allowing the user to connect to pre-Firebird-3 servers on remote machines.

If `Legacy_Auth` is allowed on the server side, you must also set the `WireCrypt` parameter to `Enabled` or `Disabled`, but not `Required`.

If `Legacy_Auth` is enabled, you will also want to change the `UserManager` setting to `Srp`, `Legacy_UserManager` (or `Legacy_UserManager, Srp` if you want to manage legacy accounts by default and/or through *gsec*). In user management statements, you can use the `USING PLUGIN` `name` clause to specify the user manager to use (only user managers listed in `UserManager` are allowed).

The `AuthServer`, `AuthClient`, `WireCrypt` and `UserManager` parameters are all set in `firebird.conf` en can be overridden per database in `databases.conf`.

Please notice: enabling `Win_Sspi` on the server activates the plugin, but doesn't grant Windows accounts any type of access to databases yet. Logging in to, say, the `employee` database without credentials (and making sure no embedded connection is made) will result in this error message:

```
SQL> connect xnet://employee;
Statement failed, SQLSTATE = 28000
Missing security context for employee
```

In other words: "We know who you are (because the `Win_Sspi` plugin identified you), but you can't come in."

The solution is to create, as SYSDBA or a co-admin, a global mapping that gives any Windows account access to databases — but no special privileges — under the same name. This is done with the following command:

```
create global mapping trusted_auth
using plugin win_sspi
from any user to user
```

`Trusted_auth` is just a chosen name for the mapping. You may use another identifier. `From any user` means that the mapping is valid for any user authenticated by the `Win_Sspi` plugin. `To user` indicates that every user will be made known under their own Windows account name in each database they connect to. If instead we had specified `to user bob`, then every Windows user authenticated by the `Win_Sspi` plugin would be `bob` in every database.

With the mapping in effect, the "Windows trusted" connection succeeds:

```
SQL> connect xnet://employee;
Database: xnet://employee, User: SOFA\PAUL
SQL> select current_user from rdb$database;

USER
==============================
SOFA\PAUL
```

With embedded connections, i.e. serverless connections handled by `Engine13`, where the client process directly opens the database file, the user is also logged in under their Windows account name if they don't provide a username when connecting. However, this doesn't require `Win_Sspi` to be enabled, nor does it need any explicit mapping:

```
SQL> connect employee;
Database: employee, User: PAUL
SQL> select current_user from rdb$database;

USER
==============================
PAUL
```

Consider whether Windows administrators should have `SYSDBA` rights

In Firebird 2.1, if the (now defunct) configuration parameter `Authentication` was *trusted* or *mixed*, Windows administrators would automatically receive SYSDBA privileges in all databases, including the security database. In Firebird 2.5 and later, this is no longer the case. This reduces the risk that administrators with little or no Firebird knowledge mess up databases or user accounts.

If you still want to apply the automatic SYSDBA mapping as it was in Firebird 2.1, login as SYSDBA and give the command:

```
create global mapping win_admin_sysdba
using plugin win_sspi
from predefined_group domain_any_rid_admins
to user sysdba
```

This grants all Windows administrators automatic SYSDBA rights in every database (including the

security database, so they can manage user accounts), provided that they are authenticated by the `Win_Sspi` plugin. To achieve this, they must connect

- without supplying any user credentials, and
- making sure that the `Engine13` provider doesn't kick in. This is easily achieved with a connection string like `xnet://local-path-or-alias`.

To give just one administrator — or indeed any user — full SYSDBA power, use this command:

```
create global mapping frank_sysdba
using plugin win_sspi
from user "sofa\frank"
to user sysdba
```

The double quotes are necessary because of the backslash in the username. (Specifying just `frank` will be accepted by Firebird, but won't result in a working mapping on most, if not all, Windows systems.)

You can drop any mapping with the command:

```
DROP [GLOBAL] MAPPING mapping_name
```

E.g.:

```
drop global mapping win_admin_sysdba;
drop global mapping frank_sysdba;
```

The `GLOBAL` keyword is necessary if it concerns a global mapping, and you're not directly connected to the security database where the mapping is registered.

5.3. Administration tools

The Firebird kit does not come with a GUI admin tool. It does have a set of command-line tools — executable programs which are located in the `bin` subdirectory of your Firebird installation (on Windows, they are in the installation directory itself). One of them, `isql`, has already been introduced to you.

The range of excellent GUI tools available for use with a Windows client machine is too numerous to describe here. At least one of them, *FlameRobin*, is also available for Linux.

Explore the following sites for more options:

- Downloads > Third-party Tools & Drivers at https://firebirdsql.org
- Download > Tools > Administration page at https://www.ibphoenix.com for more options.

Remember: you can use a Windows client to access a Linux server and vice-versa.

Chapter 6. Working with databases

In this part of the manual you will learn:

- how to connect to an existing database,

- how to create a database,

- and some things you should know about Firebird SQL.

In as much as remote connections are involved, we will use the TCP/IP protocol.

6.1. Connection strings

If you want to connect to a database or create one you have to supply, amongst other things, a *connection string* to the client application (or, if you are a programmer, to the routines you are calling). A connection string uniquely identifies the location of the database on your computer, local network, or even the Internet.

6.1.1. Local connection strings

An explicit local connection string consists of the path + filename specification in the native format of the filesystem used on the server machine, for example

- on a Linux or other Unix-like server:

```
/opt/firebird/examples/empbuild/employee.fdb
```

- on a Windows server:

```
C:\Biology\Data\Primates\Apes\populations.fdb
```

Many clients also allow relative path strings (e.g. "..\examples\empbuild\employee.fdb"), but you should use these with caution, as it's not always obvious how they will be expanded. Getting an error message is annoying enough, but applying changes to another database than you thought you were connected to may be disastrous.

Instead of a file path, the local connection string may also be a *database alias* that is defined in databases.conf, as mentioned earlier. The format of the alias depends only on how it's defined in the configuration file, not on the server filesystem. Examples are:

- zappa

- blackjack.fdb

- poker

Upon receiving a local connection string, the Firebird client will first attempt to make a direct, embedded connection to the database file, bypassing authentication but respecting the SQL

privileges and restrictions of the supplied user and/or role name. That is, if the `Engine13` provider is enabled in `firebird.conf` or `databases.conf` — which it is by default. If the database file exists, but the connection fails because the client process doesn't have the required access privileges to the file, a client-server connection is attempted (by the `Loopback` provider), in this order:

1. On Windows: using XNET (shared memory) on the local machine;

2. Using TCP/IP via `localhost`.

You can force Firebird to use a certain protocol (and skip the embedded connection attempt) by prepending the protocol in URL style:

- `inet://zappa` (TCP/IP connection using an alias on the local machine)

- `inet:///opt/firebird/examples/citylife.fdb` (TCP/IP connection using an absolute path on the local POSIX machine — notice the extra slash for the root dir)

- `inet://C:\Work\Databases\Drills.fdb` (TCP/IP connection using an absolute path on the local Windows machine)

- `xnet://security.db` (XNET connection using an alias on the local Windows machine)

- `xnet://C:\Programmas\Firebird\Firebird_3_0\security3.fdb` (XNET connection using the full path on the local Windows machine)

> If your XNET connections fail, it may be because the local protocol isn't working properly on your machine. If you're running Windows with terminal services enabled, this can often be fixed by setting `IpcName` to `Global\FIREBIRD` in the configuration file `firebird.conf` (don't forget to uncomment the parameter and restart the server).
>
> If setting `IpcName` doesn't help, and you can't get the local protocol enabled, you can usually work around the problem by using `inet://`, or putting "`localhost:`" before your database paths or aliases, thus turning them into TCP/IP connection strings (discussed below).

6.1.2. TCP/IP connection strings

Firebird has two forms of TCP/IP connection strings:

1. `{inet|inet4|inet6}://[<host>[:<port>]/]<path-or-alias>`

2. `<host>[/port]:<path-or-alias>`

With:

`<host>`
  a server name or IP address (for IPv6 addresses, enclose them in [and])

`<port>`
  port number or service name

`<path-or-alias>`

either the absolute path + filename on the server machine, or an alias defined on the server machine

Examples:

- On Linux/Unix:

```
pongo:/opt/firebird/examples/empbuild/employee.fdb
inet://pongo//opt/firebird/examples/empbuild/employee.fdb
bongo/3052:fury
inet://bongo:3052/fury
112.179.0.1:/var/Firebird/databases/butterflies.fdb
inet://112.179.0.1//var/Firebird/databases/butterflies.fdb
localhost:blackjack.fdb
inet://localhost/blackjack.fdb
```

- On Windows:

```
siamang:C:\Biology\Data\Primates\Apes\populations.fdb
inet://siamang/C:\Biology\Data\Primates\Apes\populations.fdb
sofa:D:\Misc\Friends\Rich\Lenders.fdb
inet://sofa/D:\Misc\Friends\Rich\Lenders.fdb
inca/fb_db:D:\Traffic\Roads.fdb
inet://inca:fb_db/D:\Traffic\Roads.fdb
127.0.0.1:Borrowers
inet://127.0.0.1/Borrowers
```

Notice how the aliased connection strings don't give any clue about the server OS. And they don't have to, either: you talk to a Linux Firebird server just like you talk to a Windows Firebird server. In fact, specifying an explicit database path is one of the rare occasions where you have to be aware of the difference.

6.1.3. XNET connection strings

The syntax for XNET URLs is:

`{xnet}://<path-or-alias>`

Since XNET is a purely local protocol, you can't include a hostname or port.

6.1.4. NetBEUI connection strings

Support for NetBEUI (named pipes, a.k.a. WNET) connections was removed in Firebird 5.

6.1.5. Third-party programs

Please be aware that some third-party client programs may have different requirements for the

composition of connection strings. Refer to their documentation or online help to find out.

6.2. Connecting to an existing database

A sample database named `employee.fdb` is located in the `examples/empbuild` subdirectory of your Firebird installation. It is also reachable under its alias `employee`. You can use this database to "try your wings".

If you move or copy the sample database, be sure to place it on a hard disk that is physically attached to your server machine. Shares, mapped drives or (on Unix) mounted SMB (Samba) file systems will not work. The same rule applies to any databases that you create or use.

Connecting to a Firebird database requires — implicit or explicit — authentication. To work with objects inside the database, such as tables, views and functions, you (i.e. the Firebird user you're logged in as) need explicit permissions on those objects, unless you own them (you own an object if you have created it) or if you're connected as user `SYSDBA` or with the role `RDB$ADMIN`. In the example database `employee.fdb`, sufficient permissions have been granted to `PUBLIC` (i.e. any authenticated user) to enable you to view and modify data to your heart's content.

For simplicity here, we will look at authenticating as `SYSDBA` using the password `masterkey`. Also, to keep the lines in the examples from running off the right edge, we will work with local databases and use aliases wherever possible. Of course, everything you'll learn in these sections can also be applied to remote databases, simply by supplying a TCP/IP connection string.

6.2.1. Connecting with `isql`

Firebird ships with a text-mode client named *isql* (Interactive SQL utility). You can use it in several ways to connect to a database. One of them, shown below, is to start it in interactive mode. Go to the directory where the Firebird tools reside (see Default disk locations if necessary) and type `isql` (Windows) or `./isql` (Linux) at the command prompt.

> ℹ️ In the following examples, ◀ means "hit Enter "

```
C:\Programmas\Firebird\Firebird_3_0>isql◀
Use CONNECT or CREATE DATABASE to specify a database
SQL>connect xnet://employee user sysdba password masterkey;◀
```

> ❗ • In `isql`, every SQL statement must end with a semicolon. If you hit Enter and the line doesn't end with a semicolon, isql assumes that the statement continues on the next line and the prompt will change from `SQL>` to `CON>`. This enables you to split long statements over multiple lines. If you hit Enter after your statement, and you've forgotten the semicolon, just type it after the `CON>` prompt on the next line and press Enter again.
>
> • If the connection string doesn't start with a host or protocol name, a direct serverless connection to the database is attempted. This may fail if your OS

login doesn't have sufficient access rights to the database file. In that case, connect to `localhost:path-or-alias` or specify a protocol like `xnet://` (Windows only) or `inet://`. Then the server process (usually running as user `firebird` on POSIX or `LocalSystem` on Windows) will open the file. On the other hand, network-style connections may fail if a user created the database in direct-access mode and the server doesn't have enough access rights.

> You can optionally enclose the path, the username and/or the password in single (') or double (") quotes. If the path contains spaces, quoting is mandatory. Case-sensitive usernames (created like this: `create user "Jantje" password …`) and usernames with spaces, international characters or other "`funny stuff`" also need to be double-quoted.

At this point, `isql` will inform you that you are connected:

```
Database: xnet://employee, User: SYSDBA
SQL>
```

You can now continue to play about with the `employee` database. With `isql` you can query data, get information about the metadata, create database objects, run data definition scripts and much more.

To get back to the OS command prompt, type:

```
SQL>quit;
```

You can also type `EXIT` instead of `QUIT`, the difference being that `EXIT` will first commit any open transactions, making your modifications permanent.

6.2.2. Connecting with a GUI client

Some GUI client tools take charge of composing the `CONNECT` string for you, using server, path (or alias), username and password information that you type into prompting fields. Supply the various elements as described in the preceding topic.

> - It is also quite common for such tools to expect the entire server + path/alias as a single connection string — just like `isql` does.
> - Remember that file names and commands on Linux and other Unix-like platforms are case-sensitive.

6.3. Creating a database using `isql`

There is more than one way to create a database with `isql`. Here, we will look at one simple way to create a database interactively — although, for your serious database definition work, you should create and maintain your metadata objects using data definition scripts.

6.3.1. Starting `isql`

To create a database interactively using the `isql` command shell, type `isql` (Windows) or `./isql` (Linux) at the command prompt in the directory where the Firebird tools are.

> 🛈 In the following examples, ◀ means "hit Enter"

```
C:\Programmas\Firebird\Firebird_3_0>isql◀
Use CONNECT or CREATE DATABASE to specify a database
```

6.3.2. The CREATE DATABASE statement

Now you can create your new database interactively. Let's suppose that you want to create a database named `test.fdb` and store it in a directory named `data` on your `D` drive:

```
SQL>create database 'D:\data\test.fdb' page_size 8192◀
CON>user 'sysdba' password 'masterkey';◀
```

> ⛔ • In the `CREATE DATABASE` statement it is *mandatory* to place quote characters (single or double) around path and password. This is different from the `CONNECT` statement. Quoting the username is optional, unless it is case-sensitive or contains spaces, international characters or any other character that is not allowed in a regular (unquoted) identifier.
>
> • If the connection string doesn't start with a host or protocol name, creation of the database file is attempted with your OS login as the owner. This may or may not be what you want (think of access rights if you want others to be able to connect). If you prepend `localhost:` or a protocol to the path or alias, the server process will create and own the file.

The database will be created and, after a few moments, the SQL prompt will reappear. You are now connected to the new database and can proceed to create some test objects in it.

To verify that there really is a database there, let's first type in this query:

```
SQL>select * from rdb$relations;◀
```

Although you haven't created any tables yet, the screen will fill up with a large amount of data! This query selects all rows in the system table `RDB$RELATIONS`, where Firebird stores the metadata for tables. An "empty" database is not really empty: it contains a number of system tables and other objects. The system tables will grow as you add more user objects to your database.

To get back to the command prompt type `QUIT` or `EXIT`, as explained in the section on connecting.

6.3.3. Creating a database as a non-privileged user

In Firebird 5, if you try to create a database other than in embedded mode as someone who is not a Firebird admin (i.e. SYSDBA or an account with equal rights), you may be in for a surprise:

```
SQL>create database 'xnet://D:\data\mydb.fdb' user 'john' password 'lennon';
Statement failed, SQLSTATE = 28000
no permission for CREATE access to DATABASE D:\DATA\MYDB.FDB
```

Non-admin users must explicitly be granted the right to create databases by a Firebird admin:

```
SQL>grant create database to user john;
```

After that, they can create databases.

Notice that with a serverless connection, i.e. without specifying a host name or protocol before the database name (and `Engine13` enabled!), Firebird won't deny any `CREATE DATABASE` statement. It will only fail if the client process doesn't have sufficient rights in the directory where the database is to be created.

6.4. Firebird SQL

Every database management system has its own idiosyncrasies in the ways it implements SQL. Firebird adheres to the SQL standard more rigorously than most other RDBMSes. Developers migrating from products that are less standards-compliant often wrongly suppose that Firebird is quirky, whereas many of its apparent quirks are not quirky at all.

6.4.1. Division of an integer by an integer

Firebird accords with the SQL standard by truncating the result (quotient) of an integer/integer calculation to the next lower integer. This can have bizarre results unless you are aware of it.

For example, this calculation is correct in SQL:

```
1 / 3 = 0
```

If you are upgrading from an RDBMS which resolves integer/integer division to a float quotient, you will need to alter any affected expressions to use a float or scaled numeric type for either dividend, divisor, or both.

For example, the calculation above could be modified thus to produce a non-zero result:

```
1.000 / 3 = 0.333
```

6.4.2. Things to know about strings

String delimiter symbol

Strings in Firebird are delimited by a pair of single quote (apostrophe) symbols: `'I am a string'` (ASCII code 39, *not* 96). If you used earlier versions of Firebird's relative, InterBase®, you might recall that double and single quotes were interchangeable as string delimiters. Double quotes cannot be used as string delimiters in Firebird SQL statements.

Apostrophes in strings

If you need to use an apostrophe inside a Firebird string, you can "escape" the apostrophe character by preceding it with another apostrophe.

For example, this string will give an error:

```
'Joe's Emporium'
```

because the parser encounters the apostrophe and interprets the string as `'Joe'` followed by some unknown keywords. To make it a legal string, double the apostrophe character:

```
'Joe''s Emporium'
```

Notice that this is **two** single quotes, not one double-quote.

You can also use the alternative quote string literal, allowing you to embed the quote without escaping:

```
q'{Joe's Emporium}'
```

Concatenation of strings

The concatenation symbol in SQL is two "pipe" symbols ("||", or a pair of ASCII 124 without space between). In SQL, the "+" symbol is an arithmetic operator, and it will cause an error if you attempt to use it for concatenating strings. The following expression prefixes a character column value with the string "Reported by:":

```
'Reported by: ' || LastName
```

Firebird will raise an error if the result of a string concatenation exceeds the maximum (var)char size of 32 Kb. If only the *potential* result — based on variable or field size — is too long you'll get a warning, but the operation will be completed successfully. (In pre-2.0 Firebird, this too would cause an error and halt execution.)

See also the section below, Expressions involving NULL, about concatenating in expressions involving NULL.

Double-quoted identifiers

Before the SQL-92 standard, it was not legal to have object names (identifiers) in a database that duplicated keywords in the language, were case-sensitive or contained spaces or special characters[1]. SQL-92 introduced a new syntax to make any of them legal, provided that the identifiers are defined within pairs of double-quote symbols (ASCII 34) and were always referred to using double-quote delimiters (so called quoted *or* delimited identifiers).

The purpose of this "gift" was to make it easier to migrate metadata from non-standard RDBMSes to standards-compliant ones. The downside is that, if you choose to define an identifier in double quotes, its case-sensitivity and the enforced double-quoting will remain mandatory.

Firebird does permit a slight relaxation under a very limited set of conditions. If the identifier which was defined in double-quotes:

1. is defined as all upper-case,

2. is not a keyword, and

3. conforms to the other rules of regular identifiers[1],

…then it can be used in SQL unquoted and case-insensitively. (But as soon as you put double-quotes around it, you must match the case again!)

> Don't get too smart with this! For instance, if you have tables "TESTTABLE" and "TestTable", both defined within double-quotes, and you issue the command:
>
> ```
> SQL>select * from TestTable;
> ```
>
> …you will get the records from "TESTTABLE", not "TestTable"!

Unless you have a compelling reason to define quoted identifiers, it is recommended that you avoid them. Firebird happily accepts a mix of quoted and unquoted identifiers — so there is no problem including that keyword which you inherited from a legacy database, if you need to.

> Some database admin tools enforce double-quoting of *all* identifiers by default. Try to choose a tool which makes double-quoting optional.

6.4.3. Expressions involving NULL

In SQL, NULL is not a value. It is a condition, or *state*, of a data item, in which its value is unknown. Because it is unknown, NULL cannot behave like a value. When you try to perform arithmetic on NULL, or involve it with values in other expressions, the result of the operation will almost always be NULL. It is not zero or blank or an "empty string", and it does not behave like any of these values.

Below are some examples of the types of surprises you will get if you try to perform calculations and comparisons with NULL.

The following expressions all return NULL:

- `1 + 2 + 3 + NULL`

- `not (NULL)`

- `'Home ' || 'sweet ' || NULL`

You might have expected 6 from the first expression and "Home sweet" from the third, but as we just said, `NULL` is not like the number 0 or an empty string — it's far more destructive!

The following expression:

```
FirstName || ' ' || LastName
```

will return `NULL` if either `FirstName` or `LastName` is `NULL`. Otherwise, it will nicely concatenate the two names with a space in between — even if any one of the variables is an empty string.

> Think of `NULL` as `UNKNOWN` and these strange results suddenly start to make sense! If the value of `Number` is unknown, the outcome of '1 + 2 + 3 + Number' is also unknown (and therefore `NULL`). If the content of `MyString` is unknown, then so is 'MyString || YourString' (even if `YourString` is non-NULL). Et cetera.

Now let's examine some PSQL (Procedural SQL) examples with `if`-constructs:

- Equals ('=')

```
if (a = b) then
  MyVariable = 'Equal';
else
  MyVariable = 'Not equal';
```

After executing this code, `MyVariable` will be `'Not equal'` if both a and b are `NULL`. The reason is that a = b yields `NULL` if at least one of them is `NULL`. If the test expression of an "if" statement is `NULL`, it behaves like `false`: the 'then' block is skipped, and the 'else' block executed.

> Although the expression may *behave* like `false` in this case, it's still `NULL`. If you try to invert it using `not()`, what you get is another `NULL` — not "true".

- Not equals ('<>')

```
if (a <> b) then
  MyVariable = 'Not equal';
else
  MyVariable = 'Equal';
```

Here, `MyVariable` will be `'Equal'` if a is `NULL` and b isn't, or vice versa. The explanation is analogous to that of the previous example.

The `DISTINCT` keyword comes to the rescue!

Firebird 2 and above implement IS [NOT] DISTINCT allowing you to perform (in)equality tests that take NULL into account. The semantics are as follows:

- Two expressions are DISTINCT if they have different values or if one is NULL and the other isn't;
- They are NOT DISTINCT if they have the same value or if they are both NULL.

Notice that if neither operand is NULL, IS DISTINCT works exactly like the "<>" operator, and IS NOT DISTINCT like the "=" operator.

IS DISTINCT and IS NOT DISTINCT always return `true` or `false`, never NULL.

Using DISTINCT, you can rewrite the first PSQL example as follows:

```
if (a is not distinct from b) then
  MyVariable = 'Equal';
else
  MyVariable = 'Not equal';
```

And the second as:

```
if (a is distinct from b) then
  MyVariable = 'Not equal';
else
  MyVariable = 'Equal';
```

These versions will give you the results that a normal (i.e. not SQL-brainwashed) human being would expect, whether there are NULLs involved or not.

More about `NULL`s

A lot more information about NULL behaviour can be found in the *Firebird Null Guide*, at these locations:

https://www.firebirdsql.org/file/documentation/html/en/firebirddocs/nullguide/firebird-null-guide.html (HTML)
https://www.firebirdsql.org/file/documentation/pdf/en/firebirddocs/nullguide/firebird-null-guide.pdf (PDF)

---

[1] Only a-z, A-Z, 0-9 and _ are allowed in regular identifiers, Firebird also allows $, and the first character must be a-z or A-Z

Chapter 7. Protecting your data

7.1. Backup

Firebird comes with two utilities for backing up and restoring your databases: *gbak* and *nbackup*. Both can be found in the Firebird installation directory (Windows) or its `bin` subdirectory (Linux). Firebird databases can be backed up while users are connected to the system and going about their normal work. The backup will be taken from a snapshot of the database at the time the backup began.

Regular backups and occasional restores should be a scheduled part of your database management activity.

> ⚠️ Except in nbackup's lock mode, do not use external proprietary backup utilities or file-copying tools such as *WinZip*, `tar`, `copy`, `xcopy`, etc., on a database which is running. Not only will the backup be unreliable, but the disk-level blocking used by these tools can corrupt a running database.

> ❗ Study the warnings in the next section about database activity during restores!

More information about `gbak` can be found here (HTML and PDF version, same content):

https://www.firebirdsql.org/file/documentation/html/en/firebirddocs/gbak/firebird-gbak.html
https://www.firebirdsql.org/file/documentation/pdf/en/firebirddocs/gbak/firebird-gbak.pdf

The `nbackup` manual is here (again same content in HTML and PDF):

https://www.firebirdsql.org/file/documentation/html/en/firebirddocs/nbackup/firebird-nbackup.html
https://www.firebirdsql.org/file/documentation/pdf/en/firebirddocs/nbackup/firebird-nbackup.pdf

7.2. How to corrupt a database

The following sections constitute a summary of things *not* to do if you want to keep your Firebird databases in good health.

7.2.1. Disabling forced writes

Firebird is installed with forced writes (synchronous writes) enabled by default. Modifications are written to disk immediately upon posting.

It is possible to configure a database to use asynchronous data writes — whereby modified or new data are held in the memory cache for periodic flushing to disk by the operating system's I/O subsystem. The common term for this configuration is *forced writes off* (or *disabled*). It is sometimes resorted to in an attempt to improve performance during large batch operations.

Disabling forced writes on Windows

The big warning here is: do *not* disable forced writes on a Windows server. It has been observed that the Windows server platforms do not flush the write cache until the Firebird service is shut down. Apart from power interruptions, there is just too much that can go wrong on a Windows server. If it should hang, the I/O system goes out of reach and your users' work will be lost in the process of rebooting.

Disabling forced writes on Linux

Linux servers are safer for running an operation with forced writes disabled temporarily. Still, do not leave it disabled once your large batch task is completed, unless you have a very robust fall-back power system.

7.2.2. Restoring a backup to a running database

One of the restore options in the gbak utility (`gbak -rep[lace_database]`) allows you to restore a gbak file over an existing database. It is possible for this style of restore to proceed without warning while users are logged in to the database. Database corruption is almost certain to be the result.

Notice that the shortest form of this command is `gbak -rep`, not `gbak -r` as it used to be in older Firebird versions. What happened to `gbak -r`? It is now short for `gbak -recreate_database`, which functions the same as `gbak -c[reate]` and throws an error if the specified database already exists. You can force overwriting of the existing database by adding the `o[verwrite]` flag though. This flag is only supported with `gbak -r`, not with `gbak -c`.

These changes have been made because many users thought that the `-r` switch meant *restore* instead of replace — and only found out otherwise when it was too late.

Be aware that you will need to design your admin tools and procedures to prevent any possibility for any user (including SYSDBA) to restore to your active database if any users are logged in.

If is practicable to do so, it is recommended to restore to spare disk space using the `gbak -c` option and test the restored database using `isql` or your preferred admin tool. If the restored database is good, shut down the old database (you can use the `gfix` command-line tool for this; see Firebird Database Housekeeping Utility (HTML) or Firebird Database Housekeeping Utility (PDF)). Make a filesystem copy of the old database just in case and then copy the restored database file(s) over their existing counterparts.

7.2.3. Allowing users to log in during a restore

If you do not block access to users while performing a restore using `gbak -rep` then users may be able to log in and attempt to do operations on data. Corrupted structures will result.

Chapter 8. How to get help

The community of willing helpers around Firebird goes a long way back, to many years before the source code for its ancestor, InterBase® 6, was made open source. Collectively, the Firebird community does have all the answers! It even includes some people who have been involved with it since it was a design on a drawing board in a bathroom in Boston.

- Visit the official Firebird Project site at https://www.firebirdsql.org and join the user support lists, in particular `firebird-support`. Look at https://www.firebirdsql.org/en/mailing-lists/ for instructions.

- Use the Firebird documentation index at https://www.firebirdsql.org/en/documentation/.

- Visit the Firebird knowledge site at https://www.ibphoenix.com to look up a vast collection of information about developing with and using Firebird. IBPhoenix also sells a Developer CD with the Firebird binaries and lots of documentation.

- Order the official three-volume *Firebird Book*, *Second Edition* at https://www.ibphoenix.com/products/books/firebird_book, for more than 1200 pages jam-packed with Firebird information. (*Notice:* at the time of this writing, the *Firebird Book* is not yet up-to-date with Firebird 3 and higher.)

- Read the Release Notes for your Firebird version!

Chapter 9. How to give help

Firebird exists, and continues to be improved, thanks to a community of volunteers who donate their time and skills to bring you this wonderful piece of software. But volunteer work alone is not enough to keep an enterprise-level RDBMS such as Firebird up-to-date. The Firebird Foundation supports Firebird development financially by issuing grants to designers and developers. If Firebird is useful to you, and you'd like to give something back, please visit the Foundation's pages and consider making a donation or becoming a member or sponsor.

Chapter 10. The Firebird Project

The developers, designers and testers who gave you Firebird and several of the drivers are members of the Firebird open source project at Google Groups, SourceForge and GitHub. The Firebird SourceForge address is https://sourceforge.net/projects/firebird/. The sources and issue tracker are hosted on GitHub: https://github.com/FirebirdSQL/firebird. At that site are the source code tree, the download packages and a number of technical files related to the development and testing of the code bases.

The Firebird Project developers and testers use an email list forum — firebird-devel Google Group — as their "virtual laboratory" for communicating with one another about their work on enhancements, bug-fixing and producing new versions of Firebird.

Anyone who is interested in watching their progress can join this forum. However, user support questions are a distraction which they do not welcome. Please do not try to post your user support questions there! These belong in the firebird-support Google Group.

Happy Firebirding!

Appendix A: Document History

The exact file history is recorded in the firebird-documentation git repository; see https://github.com/FirebirdSQL/firebird-documentation

Revision History

0.2	08 Nov 2023	M R	Remove mention that `ServerMode` is per-database configurable (it's not)
0.1	07 Nov 2023	M R	• Copied *Firebird 3 Quick Start Guide* as the starting point for *Firebird 5 Quick Start Guide* • Restarted the version numbering and cleared the document history, for older revision history, see *Firebird 3 Quick Start Guide* • Replaced references to Firebird 3 with Firebird 5 where applicable • Replaced `qsg3` section prefix with `qsg5` • Removed UDF directory from *disk locations* • Removed references to WNET/NetBEUI protocol • Added security database upgrade instructions from Firebird 3 and higher • Updated SQL user management statement syntax to be more complete • Merged URL-style and TCP/IP protocol description • Added replacement images for Installer and Services on Windows • Installer no longer offers option to enable legacy authentication • Various copy-editing and textual tweaks

Appendix B: License notice

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the "License"); you may only use this Documentation if you comply with the terms of this License. Copies of the License are available at https://www.firebirdsql.org/pdfmanual/pdl.pdf (PDF) and https://www.firebirdsql.org/manual/pdl.html (HTML).

The Original Documentation is titled *Firebird 5 Quick Start Guide*. This Documentation was derived from *Firebird 3 Quick Start Guide*.

The Initial Writer of the Original Documentation is: IBPhoenix Editors.

Copyright © 2002-2004. All Rights Reserved. Initial Writer contact: hborrie at ibphoenix dot com.

Contributors: Paul Vinkenoog, Mark Rotteveel.

Portions created by Paul Vinkenoog are Copyright © 2004-2016. All Rights Reserved. Contributor contact: paul at vinkenoog dot nl.

Portions created by Mark Rotteveel are Copyright © 2020-2023. All Rights Reserved. Contributor contact: mrotteveel at users dot sourceforge dot net.

Alphabetical index